メタ戦略アルゴリズムに対するロバストな並列化 (計算機科学基礎理論の新展開)

AUTHOR(S):

石橋, 正裕; 小野, 廣隆; 朝廣, 雄一; 山下, 雅史

CITATION:

石橋, 正裕 ...[et al]. メタ戦略アルゴリズムに対するロバストな並列化 (計算機科学基礎理論の新展開). 数理解析研究所講究録 2003, 1325: 1-7

ISSUE DATE:

2003-05

URL:

http://hdl.handle.net/2433/43164

数理解析研究所講究録 1325 巻 2003 年 1-7

メタ戦略アルゴリズムに対するロバストな並列化

石橋 正裕 (Masahiro Ishibashi) *　　小野 廣隆 (Hirotaka Ono) †
朝廣雄一 (Yuichi Asahiro) ‡　　山下 雅史 (Masafumi Yamashita) †

* 九州大学大学院システム情報科学府　† 九州大学大学院システム情報科学研究院
*† Deptartment of Computer Science and Communication Engineering,
Graduate School of Information Science and Electrical Engineering,
Kyushu University

‡ 九州産業大学情報科学部社会情報システム学科
‡ Deptartment of Social Information Systems, Faculty of Information Science,
Kyushu Sangyo University

1　はじめに

近年，NP 困難な組合せ最適化問題に対する高精度近似解法であるメタ戦略が，その有用性から盛んに研究されている [3]．多くのメタ戦略は局所探索法に基づいており，現実的な時間で精度の高い近似解を求められることが知られている．一方，メタ戦略の発展に伴い，メタ戦略を複数の計算機で並列化することで，より高精度で高速な解法を開発しようとする研究も盛んに行われている [5]．メタ戦略を並列化する場合，対象とするメタ戦略を熟知した上でパラメータチューニングや実験的な作り込みが必要とされる場合が多い．しかし，このような並列化手法では，一般に個々のメタ戦略のデータ構造，動作の理解に多くの労力が必要となる．また，実際に実装して動かしてみるまでその性能が分からないため，使用計算機数を増やしても期待通りの性能が得られないなどの問題点がある．そこで本研究ではメタ戦略の並列化プロセス自体に着目し，与えられたメタ戦略アルゴリズムからいかにして要求 (解精度，実行時間) に見合った並列メタ戦略を設計するか，その方法論を提案，議論する．このため，まずメタ戦略の「粒度」を並列化の視点から定義する．本設計手法はこの粒度の観点から対象とするメタ戦略の性能を実験的に解析することにより，性能見積もり付の並列メタ戦略を設計する．

2　メタ戦略

2.1　局所探索法

局所探索法はある解 σ に少しの変形を加えることで得られる解の集合 $\mathrm{N}(\sigma)$(近傍と呼ばれる) 内を探索し，評価関数 f の下で改善解を発見するとその改善解の近傍から同様に探索を行い，近傍内に改善解が見つからなくなるまでこれらの操作を繰り返す手法である．局所探索法のアルゴリズムを次に示す．

アルゴリズム 局所探索法 $\mathrm{LS}(N, f, \sigma)$

(1): $f(\sigma') < f(\sigma)$ を満たす解 $\sigma' \in N(\sigma)$ が見つかれば，$\sigma := \sigma'$ とし，この操作を繰り返す．

(2): σ を出力し終了．

2.2　メタ戦略アルゴリズム

ここでは多くのメタ戦略を捉えることのできる枠組みでアルゴリズムを記述する．近傍集合 Π，初期解 σ，評価関数 f，目的関数 g，解生成戦略 S，初期探索パラメータ α_0，初期状態 q_{init}，終了状態 q_{halt}，状態遷移関数 δ が与えられたときの一般的なメタ戦略のアルゴリズムは次の通りである．

アルゴリズム メタ戦略アルゴリズム
$\mathrm{MH}(\Pi, \sigma, f, g, S, \alpha_0, q_{init}, q_{halt}, \delta)$

(1): 暫定解 σ_{best}:=σ，状態 q:=q_{init}，探索パラメータ α:=α_0 とする．

(2): σ^*:=σ として解のコピーを行う．

(3): 全ての近傍 $N_i \in \Pi$ に対して改善解が発見できなくなるまで σ':=$LS(N_i, f, \sigma)$, σ:=σ' として局所探索を行う．局所探索中に $g(\sigma'_{best}) < g(\sigma_{best})$ となる解 σ'_{best} を見つけたら，σ_{best}:=σ'_{best} とする．

(4): 探索内容よりパラメータ α を更新する．

(5): $f(\sigma^*) > f(\sigma')$ ならば σ:=σ', q:=$\delta(q, \sigma, \alpha)$ として次状態へ．

(6): $q = q_{halt}$ ならば σ_{best} を出力して終了．

(7): 解生成戦略 $S(q)$ で解 σ を生成し (2) へ．

2.3 粒度

次にメタ戦略の「粒度」を定義する．

シーケンス: 初期状態 q_{init} ～終了状態 q_{stop}

フェーズ : 状態 q ～次状態 q:= $\delta(q, \sigma, \alpha)$

トライアル: 全ての $N \in \Pi$ に対する局所探索 $LS(N, f, \sigma)$ が終了するまで

サーチ : ある $N \in \Pi$ に対する局所探索 $LS(N, f, \sigma)$ が終了するまで

粒度を示した図が図 1である．粒度の定義より 2 つの並列化手法が考えられる．一つ目は各計算機にそれぞれ異なる初期解を与えて並列に独立なメタ戦略を行うシーケンス並列化である．二つ目はフェーズ中のトライアルを各計算機で分担するトライアル並列化である．サーチ粒度での並列化も考えられるが，一般には対象が非常に細かくなり膨大な通信回数になることが予想されるため今回は扱わない．また，フェーズ粒度については，状態遷移が一本道になっており並列化の余地がないため扱わない．

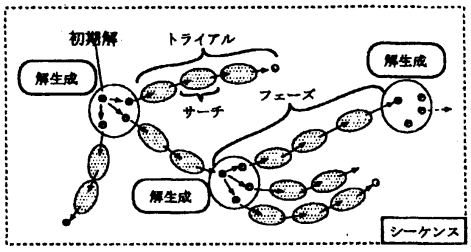

図 1: メタ戦略アルゴリズム

3 並列化

3.1 並列化のモデル

本稿では，分散環境として並列計算機よりも安価で導入しやすい PC クラスタ [1] を採用する．実現可能な分散化のメリットとして，高速化，解精度の向上 (高精度化)，メモリの分散，負荷分散などが挙げられるが，本研究では高速化と高精度化の 2 点を目的とする．PC クラスタは非同期システムで共有メモリを持たないため，各 PC 間での情報のやり取りにはメッセージ通信を用いて行う．この分散環境下での制約のため，通信には相応のコスト (通信時間) がかかる．また本研究では，マスタが各々のスレーブに対してプロセスを分配し，スレーブは割当てられたプロセスを実行するというマスタ・スレーブ方式を用いる．

3.2 シーケンス並列化

シーケンス並列化は探索開始から終了までのシーケンスを一つの粒度として扱った並列化手法である．図 2はシーケンス並列化のイメージ図である．マスタは各スレーブにそれぞれランダムに生成した解を送り，スレーブからの暫定解の受信を全スレーブがそれぞれメタ戦略を終えるまで行う．スレーブはマスタから受け取った解からメタ戦略を終了状態まで行い，探索中に暫定解を発見した時点でマスタに逐一暫定解を送信する．シーケンス並列化では，複数回のメタ戦略が同時に行われるためスレーブ台数を増やせば増やすほど良い解が見つかり易くなる．しかし，各計算機によるメタ戦略の動作と 1 台でのメタ戦略のそれは異ならないため，本来のメタ戦略で発見できる解よりも良い解は望めない．スレーブの集合を P として，シーケンス並列化のアルゴリズムは次の通りである．

アルゴリズム シーケンス並列化 – マスタ

$SP_{master}(\Pi, f, g, S, \alpha, q_{init}, q_{halt}, \delta, P)$

(1): スレーブを $|P|$ 台起動する．

(2): 各スレーブにそれぞれランダムに生成した解を送信し，$g(\sigma_{best}) = +\infty$ とする．

(3): メッセージ M_{halt} をスレーブ台数分受信するまで，スレーブから解 σ'_{best} を受信して $g(\sigma'_{best})<g(\sigma_{best})$ ならば，$\sigma_{best}:=\sigma'_{best}$ とすることを繰り返す．

(4): σ_{best} を出力して全スレーブを停止する．

アルゴリズム シーケンス並列化 – スレーブ

$SP_{slave}(\Pi, f, g, S, \alpha, q_{init}, q_{halt}, \delta)$

(1): マスタから初期解 σ を受信する．

(2): $MH(\Pi, \sigma, f, g, S, \alpha, q_{init}, q_{halt}, \delta)$ を行い，探索中に新しい暫定解 σ'_{best} を見つける度にマスタに σ'_{best} を送信．

(3): 終了メッセージ M_{halt} をマスタに送信して終了．

シーケンス並列化でのマスタ・スレーブ間の通信回数は、マスタが初期解を送るときのスレーブ台数回と，スレーブからマスタに解を送るときのスレーブ台数 × 暫定解更新回 (一般にはこれは小さい) であると考えられるので，シーケンス並列化での通信コストは非常に小さい．また，マスタの負担が少ないため，台数をかなりの数まで増やすことができる．

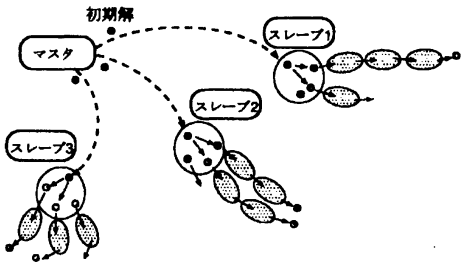

図 2: シーケンス並列化

3.3 トライアル並列化

通常メタ戦略は各フェーズにおけるトライアルを複数回行う．この性質を利用し，トライアルを複数台の計算機で同時に行うトライアル並列化により高速化が実現できる．また，各フェーズで少なくともスレーブ台数分のトライアルを行うことになり，より広い領域を探索することができるので，元のメタ戦略以上の解精度を得る可能性も考えられる．図3にトライアル並列化のイメージ図を載せる．トライアル並列化ではマスタの持つ解 σ と状態 q から解生成を行い，生成した解をスレーブに送信する．マスタはスレーブから解 σ' やその他の探索情報を受け取り，$f(\sigma_{prime}) < f(\sigma)$ ならば，次状態に遷移させる．そうでないなら状態 q のままで解生成を行い，スレーブに返すことを終了状態まで行う．スレーブはマスタから受け取った解をもとに，トライアルを行い，途中で発見した暫定解や最終的に得られた解をマスタに送信する．トライアル並列化では，1回のトライアルに対して2回の通信が必要になるため，探索中のトライアル数が多い場合には通信コストは比較的大きくなる．トライアル並列化のアルゴリズムは次の通りである．トライアルの部分を $Trial(\Pi, \sigma, f, g, \alpha)$ と表わすことにする．

アルゴリズム トライアル並列化 – マスタ

$TP_{master}(\Pi, \sigma, f, g, S, \alpha, q_{init}, q_{halt}, \delta, P)$

(1): $\sigma_{best}:=\sigma$，$q:=q_{init}$，$\alpha:=\alpha_0$ とする．

(2): スレーブ台数分の解 $\sigma:=S(q)$ を生成し，解とパラメータ $(\sigma, \sigma_{best}, \alpha)$ を各スレーブに送信する．

(3): スレーブ $p \in P$ から解 $(\sigma', \sigma'_{best})$ を受信し，$g(\sigma'_{best})<g(\sigma_{best})$ ならば $\sigma_{best}:=\sigma'_{best}$ とする．

(4): 探索内容よりパラメータ α を更新する．

(5): $f(\sigma) > f(\sigma')$ であるなら，$\sigma:=\sigma'$，$q = \delta(q, \sigma, \alpha)$ として次状態へ．

(6): $q=q_{halt}$ ならば，σ_{best} を出力して終了．

(7): 解生成戦略 $S(q)$ より解 σ を生成し，スレーブ p に $(\sigma, \sigma_{best}, \alpha)$ を送信し (5) へ．

アルゴリズム トライアル並列化 – スレーブ

$TP_{slave}(\Pi, f, g)$

(1): マスタから解とパラメータ $(\sigma, \sigma_{best}, \alpha)$ を受け取る．

(2): $\sigma:=Trial(\Pi, \sigma, f, g, \alpha)$ を行い，探索中 $g(\sigma'_{best})<g(\sigma_{best})$ となる解 σ'_{best} を見つけたら，$\sigma_{best}:=\sigma'_{best}$ とする．

(3): マスタに解 (σ, σ_{best}) を送信．

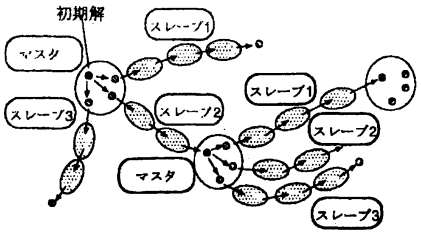

図 3: トライアル並列化

4 並列メタ戦略の見積もり

4.1 メタ戦略の性能見積もり

4.1.1 暫定解の収束解析

メタ戦略では探索が終了することがないため，終了条件を指定する必要がある．終了条件を実行時間とすることが多いが，実行時間と解精度はトレードオフの関係を持つため，単純に解精度による比較，実行時間による比較ができない．このためここでは，どのような解をどの程度の時間で見つけることができるかでメタ戦略の性能を評価することを考える．

最小化問題に対するメタ戦略ならば，最適解の下界 (発見されていれば最適解自身) にできるだけ近い解を，できるだけ高速に見つける性能が要求される．そこで並列化の対象とするメタ戦略の性能を，探索中に発見される暫定解とその解を得た時間により評価する．図 4はある問題に対するメタ戦略を 500 秒でそれぞれ *seed* を変えて 10 回実行し，横軸を時間 (秒)，縦軸を解の目的関数値と下界との誤差として実験結果を表わしたものである．この例ではいずれの探索も探索開始から数十秒で急速に暫定解を発見し，下界の 1.001 倍の目的関数値を持つ解に辿り着き，300 秒以降は新しい暫定解の発見が跡絶えているのが分かる．この最終的に得られた暫定解を収束解と呼び，収束解を発見した時間を収束時間と呼ぶ．一方，*seed* により収束解とその収束時間にもばらつきが見られるため，暫定解発見の平均的な振る舞いで評価することを考える．各時間における暫定解 10 個分の平均を探索中の各時間において求め，それぞれの暫定解の平均値により，平均収束解とその収束時間が得られる．

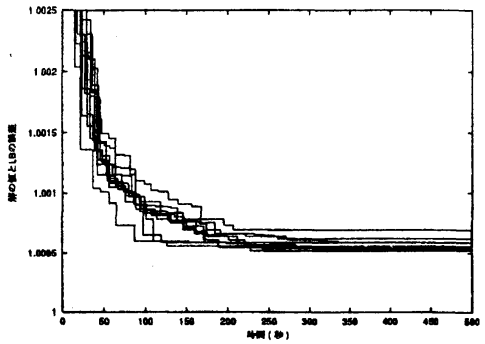

図 4: 暫定解の収束

4.1.2 暫定解の分布解析

あるメタ戦略の暫定解の収束を調べ，最終的に得られる解にばらつきが見られた場合，その暫定解の分布を調べることで対象とするメタ戦略の性能を見積もることを考える．これは最終的に得られた解を標本とし，その標本分布から母集団の分布を推定する統計的推定により求める．ここで母集団は実行可能解の集合であり，その母数や母集団の分布を知ることは難しいため，推定は点推定という手法で行う．点推定は，母集団を正規母集団と考えて標本平均と標本分散から正規母集団を推定する手法である．ここで，*seed* を変えて k 回の実行を行い，得られた暫定解が目的関数値 x_i $(i = 1, 2, \ldots, k)$ を持つ場合，

$$\text{標本平均} \quad \bar{x} = \frac{1}{k}\sum_{i=1}^{k} x_i$$

$$\text{標本分散} \quad s^2 = \frac{1}{k}\sum_{i=1}^{k}(x_i - \bar{x})$$

と表わすことができ，この標本平均 $\bar{x}$ と標本分散 s^2 をそれぞれ平均，分散とする正規分布 $N(\bar{x}, s^2)$ を推定母集団の分布とする．図 5の実線は図 4で最終的に得られた暫定解の分布を，横軸に暫定解の目的関数値と下界との誤差，縦軸に暫定解の得られた割合として表わした図である．また，図 5の点線は実線の暫定解の分布を標本として点推定を行い，得られた推定母分布である．この推定分布を発見が期待される分布として並列メタ戦略の性能評価に用いる．

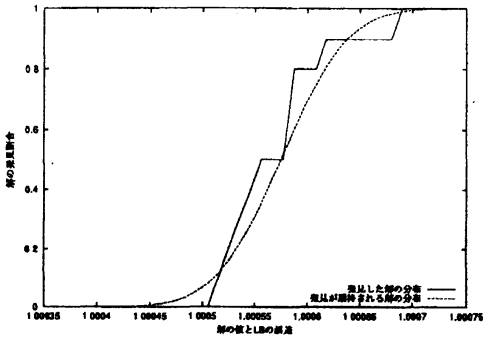

図 5: 暫定解の分布と発見が期待される解分布

4.2 シーケンス並列化の見積もり

対象とするメタ戦略から得られた暫定解の収束解析，分布解析により，シーケンス並列化の性能見積もりを行う．収束解析で得られたデータからある時間 t における暫定解の分布が得られる．各時間 t での分布を $F_t(x)$ とすると，2台でのシーケンス並列化で期待される解の分布 $F_t^{(2)}(x)$ はそれぞれ独立な解の確率変数 X_1, X_2 を用いて $F_t^{(2)}(x)= P(\min\{X_1, X_2\} \leq x)$ である．これは *seed* を変えたメタ戦略の動きが独立であるとすると $1-(1-F_t(x))^2$ に等しいので，n 台でのシーケンス並列化の性能見積もり $F_t^{(n)}(x)$ は $1-(1-F_t(x))^n$ で表わすことができる．

4.3 トライアル並列化の見積もり

4.3.1 並列度解析

トライアル並列化を行う場合，各フェーズにおけるトライアル数を調べることから始める．各フェーズにおけるトライアル数が使用可能計算機数よりも多い場合，効率の良い高速化を望むことができる．また，各フェーズにおけるトライアル数が少ない場合には，各フェーズで少なくとも使用台数分のトライアルを行うことになり探索領域が広がるため，未発見の解を見つける可能性が増えることが考えられる．以上から，並列度解析を行うことで，高速化と高精度化の並列化の方針を決めることができる．ここで，高精度化は同じ終了時間内に高精度な解を発見することができることを意味するので，対象とするメタ戦略での収束解を早い時間で見つけることのできる高速化の意味も持つ．図6はある問題に対するメタ戦略について，各フェーズにおけるトライアル数を表わしたものである．探索開始のフェーズのインデックスを1，として順に各フェーズ毎のトライアル数を縦軸に取っている．図6より，並列度は多くて150程度で，平均は約4であり，トライアル数が1のフェーズが7割を占める．これより，このメタ戦略には台数幅解析を行い，高精度化の影響を調べる必要があると言える．

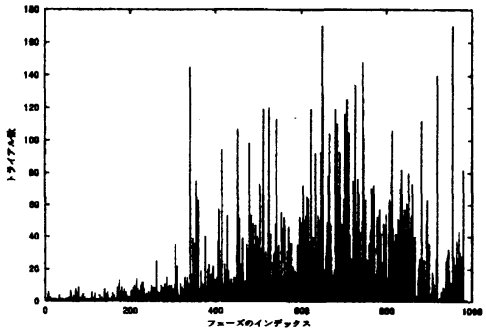

図 6: 各フェーズにおけるトライアル数

4.3.2 台数幅解析

台数幅解析では n 台でのトライアル並列化の影響を調べるために，各フェーズでのトライアルを少なくとも n 台行うことで，トライアル並列化の性能をシミュレートする．図??はある問題に対するメタ戦略に各フェーズのトライアルを最低10回は行うという制約を加えたもので，元のメタ戦略よりも収束に時間がかかるが，元のメタ戦略よりも良い解に辿り着いている．このことはトライアル並列化による高精度化が可能であることを意味する．図6の例では各フェーズでのトライアル数が1であるものが多いため，少なくともトライアルを10回以上行う場合，本来のメタ戦略に要した時間の最大10倍の時間で実験を行ったものが，並列化のより正確な見積もりになると考えられる．

4.3.3 パラメータ更新頻度解析

トライアル並列化をする際に問題となるのは，探索パラメータをどのように扱うかである．探索パラメータの更新されるタイミングによっては元のメタ戦略に忠実なパラメータ更新ができない場合がある．元のメタ戦略で探索パラメータの更新がフェーズ毎に行われる場合，マスタはパラメータが更新される度に各スレーブに送信すれば良い．しかしパラメータの更新頻度がトライアル毎である場合は単純ではない．本来のメタ戦略でi番目のトライアルt_iでのパラメータをα_iとすると，α_{i+1}はt_iの探索内容とα_iによって調節が行われる．これに対してトライアル並列化はパラメータの調整に次の2手法を採用している．k台での並列化の場合，一つはトライアル$t_i, t_{i+1}, \ldots, t_{i+k}$を同時に同じパラメータ$\alpha_i$で行う手法で1つのフェーズは固定．また，パラメータを重要視並列化では，トライアルが終わる度にマスタがそのトライアルの内容からパラメータを更新し全てのスレーブにブロードキャストする方法も考えられる．これらのパラメータ更新のタイミングが異なる動作をシミュレートするのがパラメータ更新頻度解析である．本解析では，元のメタ戦略に手を加え，パラメータの更新タイミングを (1) フェーズ粒度，(2) サーチ粒度に変更したものをそれぞれ実装して解の収束と分布を調べる．トライアル並列化はこの解析の結果性能の良かったタイミングで実装を行う．

4.4 解析からの見積もり

以上の3つの解析結果からトライアル並列化の見積もりを行う．まず (1) パラメータ更新頻度解析により決定した粒度でパラメータの調節を行い，(2) 各フェーズで使用台数分以上のトライアルを行う台数幅解析を組み込んだメタ戦略で実験を行う．さらに (3) 高速化の見積もりを行った結果得られたのが，図??である．高速化は，元のメタ戦略の並列度解析で得られた各フェーズ毎のトライアル数を台数分で畳み込み，高速化の割合を算出したものに，総トライアル数 $\times 1$ 回の送受信にかかる時間を組み込むことで見積もる．

5 実装と実験結果

本実験では，一般化割当問題 [4] に対するタブーEC法 [2] を対象としてその解析からそれぞれの手法での並列タブーEC法の見積もりを行い，実装した結果との比較を行う．一般化割当問題は代表的な組合せ最適化問題の一つであり，ベンチマークとして様々な種類のインスタンスが公開されている (http://www-or.amp.i.kyoto-u.ac.jp/members/yagiura/gap/)．実験にはこのうち，d401600という比較的大規模なサイズのインスタンスを使用する．実験環境はCPU: Pentium4 2.26GHz,MEM: 512MB,OS: Debian2.20, コンパイラ: gcc2.95.4, 通信ライブラリ: pvm3.4[1] である．使用計算機台数は2～16台でそれぞれにおいてシーケンス並列化，トライアル並列化の見積もりと実装して得た実験結果から，その高速化と高精度化の比較を行う．実験はそれぞれ1回500秒で *seed* を変えて10回ずつ行う．高速化はメタ戦略の平均収束解 (LBの1.000575倍の目的関数値を持つ解) を平均収束時間 (337秒) の何倍早い時間で見つけることができるか，で評価する．高精度化は平均収束解とLBの誤差 (0.000575) を1として，何倍LBに近づいたかで評価を行う．図7，図8はそれぞれシーケンス並列化と見積もりの高速化，高精度化の比較，図9，図10はそれぞれトライアル並列化と見積もりの高速化，高精度化の比較である．図は横軸にスレーブ台数，縦軸に台数効果を取ったものである．高速化はシーケンス並列化，トライアル並列化共に良い見積もりが得られていることがわかる．一方，高精度化についても台数が増えるにつれて若干誤差が大きくなっているものの，両者とも見積もり値は実装値に比較的近い値となっている．

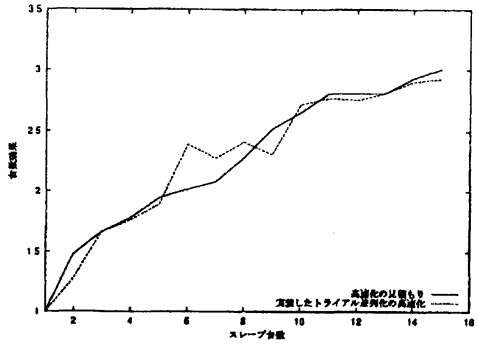

図 7: シーケンス並列化の見積もりと実装値
(高速化による比較)

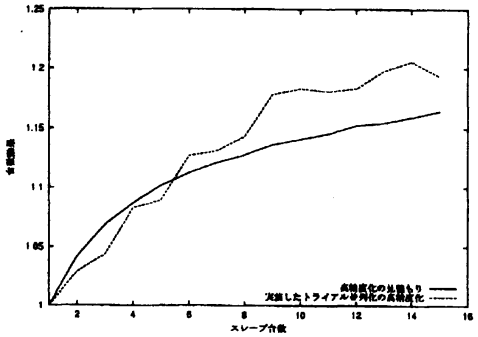

図 8: シーケンス並列化の見積もり値と実装値
(高精度化による比較)

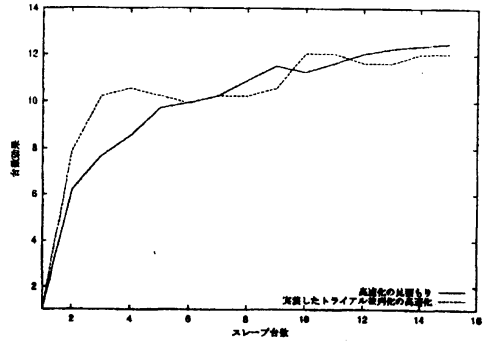

図 9: トライアル並列化の見積もりと実装値
(高速化による比較)

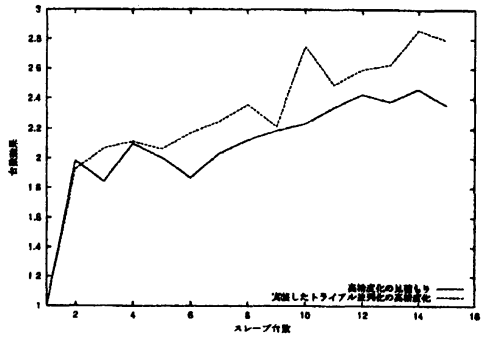

図 10: トライアル並列化の見積もりと実装値
(高精度化による比較)

6 おわりに

今回，一般のメタ戦略に適応できる「粒度」の枠組みと，粒度に基づいた二つの並列化手法の実験的解析による性能見積もり方法を提案た．また，一般化割当問題に対するタブーEC法について計算実験を行い，その見積もりが良い見積もりであることを確認した．今後は，ある問題に対する専用並列メタ戦略アルゴリズムと本並列化手法に基づく並列アルゴリズムでの性能比較を行う予定である．

参考文献

[1] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek and V. Sunderam. PVM: parallel virtual machine - a users' guide and tutorial for networked parallel computing, MIT Press, 1994.

[2] M. Yagiura, T. Ibaraki and F. Glover. An Ejection Chain Approach for the Generalized Assignment Problem, INFORMS Journal on Computing, 1999.

[3] M. Yagiura and T. Ibaraki. On metaheuristic algorithms for combinatorial optimization problems, Systems and Computers in Japan, 32, Issue 3, 33-55, 2001.

[4] S. Shani and T. Gonzalez. P-complete approximation problems, J. ACM, 23, 555-565, 1976.

[5] V.-D. Cung, S. L. Martins, C. C. Ribeiro, and C. Roucairol. Strategies for the parallel implementation of metaheuristics, Essays and Surveys in Metaheuristics, 263-308, 2001.